Design and Efficiency Analysis of a New Protocol for Malicious Hosts Detection

Mohammed Chihoub and Ramdane Maamri

Mentouri University, Constantine, Algeria, Computer Science Department
Laboratory Lire, Route Ain-El-Bey 25000, Constantine, Algeria
mchihoub@yahoo.com, rmaamri@yahoo.fr

Abstract

Mobile agents' paradigm wide acceptance depends on the ability to overcome its major problem of security. In this context, the major concern is the subtle issue of malicious hosts. Many solutions had been proposed in this area, varying from tamper free hardware to pure software protocols. Nevertheless, neither of them is fully satisfactory. This paper investigates the problem on a novel basis. It develops an incremental protocol giving the opportunity to appreciate the benefit of making weaker hints available (.i.e., information about the lower suspicious character of malicious hosts). Upon these hints and progressively, the protocol can unambiguously confirm the stronger malicious character of hosts by a process of truth enforcement. This paper develops the issues raised by the protocol and validates the protocol efficiency in regard to the weaker hints use to verify the veracity of what we have called explicitly the lower suspicious character of a host.

Keywords: *Mobile Agent's Security, Efficiency, Incremental, Malicious Hosts, Truth*

1. Introduction

In general, the objective of using autonomous mobile agents in a network environment, from the World Wide Web to the Data Grid, is clearly to search, i.e., to locate some required "items" (e.g., information, resource...) in the environment [1]. The use of autonomous roving agents to accomplish some specified tasks, on the behalf of an agent owner, had discarded the need to maintain a permanent connection during mobile agent journey. Within this assertion, the mobile agents' model outperforms the paradigm of client/server both at [2]:

1. Coping seamlessly with the transient nature of channels,

2. Producing a better overall performance, especially at low bandwidth and low reliability networks.

This model presents many advantages. However, its major drawback can be found in the fact that, no online supervision can anymore be done. Thus, the visited hosts have a total control over their visiting agents. The assumption that hosts are harmless is only not realistic in an open network. Visited hosts can severely damage mobile agents. Protecting an agent from "hosts' attacks" has become a pressing security concern. Attacking hosts leaving no such evidence of their harmful actions is known as the problem of malicious hosts. This subtle issue is by far the most difficult facet of mobile agents' security concern. Many solutions had been proposed in the area of mobile

agents' trustworthiness, varying from tamper free hardware to pure software protocols. However, neither of them is fully satisfactory [10], [11].

Our paper addresses the topic of mobile agents' trustworthiness on a novel basis. The proposed solution proceeds incrementally. It makes a clear distinction between the lower suspicious character derived from weak hints and the stronger malicious one of a host. This latter is unambiguously confirmed by the ultimate stage of truth enforcement named mediation. In essence, we are exploiting principles from the era of distributed systems investigations.

These principles are related to scarce resources such as CPU cycles and storage. Due to Barak [3], "any server whose load is proportional to the size of the system is destined to be clogged once the system grows beyond a certain size''. Therefore, the truth stability property can be used to distinguish between an honest and a malicious host. Expressing truth does not require any kind of any provision. However, a malicious host, perhaps, will keep trace of previous information. This latter is used for example, to maintain multiple answers coherences. Enforcing truth for a malicious host should use a memory and a processor intensive approach. Our desire is to overload the host, pushing it rapidly to the limit of a clogged state. We could hope that before becoming faulty (i.e., clogged), the malicious host will not be able to give coherent answers due to scarce resources capacity lack. Then, we can say that the interrogated host is becoming amnesic.

Most approaches improving agents' owner survivability are focusing on ways that detect if malicious acts have been attempted. They rely on a Third Trust Party to solve potential dispute. We believe that the impartial party should have more prerogatives and promoted to have a mediation role where more flexibility is gained.

In our opinion, splitting the characterization of a host into two different distinguishable aspects, the inexpensive suspicion property, and the stronger malicious character, should enable a better care of malicious hosts' problem. In fact, we may feel free to use any suitable protocol to construct the proof ground characterizing the low suspicious behavior. It must not be as accurate as it should be. Later on, the proof ground will be handed on to the stronger mediation activity where the false allegations are declined or the malicious acts are confirmed.

The paper is organized in six sections. In the second section, some related works are described with the objective to show the benefit of our approach. The third section shows the main contribution of this work. In the fourth section, we will develop an inexpensive suspicion detection protocol. Thereafter, it explains the mediation principles and gives our protocol proposal, and answers to practical questions. To comfort our proposed protocol, we have chosen to devote the last section to hosts' suspicious character detection efficiency validation experiment i.e., weaker hints soundness based on a statistical analysis approach. We will conclude, in the last section, by pointing out aspects of future work.

2. Related Works

With regards to the malicious hosts' problem, techniques that make tampering impossible or more difficult such as mobile cryptography, tamper free hardware, and mutating agents as in the time limited black box are less viable than those focusing on ways detecting tampering acts [4], [5], [9], [6]. When the agents' owner is provided with improper information, it is almost under some form of information assault. If the assaults can be thwarted then the survivability of the agent owner is increased [7].

Albeit, our suspicious character detecting protocol shares the key ideas presented both in Vigna's and Esparsa's schemes [8], [10], it proceeds differently.

Vigna privileged the idea of establishing unaltered initial and final states at each agent's hop. The idea is concretized via the redundancy of the hashed intermediate traces and playing fair rules enforcement. Intermediate traces key information is black statements values, i.e., system calls results. Consequently, tampering would be checked by the re-executing simulation of the agent with respect to the unaltered intermediate traces and any disparity will confirm host's cheating. Vigna's protocol accuracy is paid at the expense of excessive storage demand, increased and complicated protocol stages to implement non repudiation (i.e., fair play), and simulated re-execution of the agent.

Esparsa took another direction. His key idea is to limit the execution time of the roving agent at each host. The protocol begins by setting up a general time reference. Since truly synchronized hosts' time is unfeasible, a transmission time evaluator is activated to estimate:

1. The difference between the general time reference and the time reference of each host,

2. The transmission delay between consecutive hosts in the agent's itinerary.

Yet, an execution time estimator is also needed to calculate the necessary time for each host to execute the agent completely. According to this protocol, an honest host would exhibit values of the real execution time, the real transmission time which are confined respectively within the limit of:

1. The estimated execution time,

2. The difference between its arrival time at the new host and its finalizing time in the predecessor host.

Obviously, the absolute finalization time at a host cannot be but less than the absolute arrival time at its successor. The crucial point of the protocol is its time estimating components. When the estimation is static, better estimated times would be abstracted from extensive pre-simulation. Sophisticated real time estimation will incur additional overhead and require a permanent connection of the origin host until the transaction finishes. Hence, the off-line property is violated.

In our approach, we also care about black statements and the agent's time spent in a host. However, unlike:

1. Vigna, we consider black statements the source of irregularities in execution time. Therefore, the execution time can be settled down by ignoring the time devoted to such statements. Removing the irregularities would make the processing power of any host proportional to its load.

2. Esparza, we discard the need to resort to the general time reference and the estimating components. Rather, we consider the first host as the referential model of execution. In reality, even when the first host could cheat, our protocol would detect such acts. Our protocol does not require more than a host's signed instance of time with load at arrival, departure, and black statements boundaries. As we can see, there is no need for a permanent connection with the agent owner during the agent whole journey. The obvious benefit is the off-line property preservation

3. Major Contributions

Our proposed strategy lies within techniques detecting tampering attempts. To the best of our knowledge, no published solution has taken that direction for malicious hosts' problem exploration. Besides the truth enforcement mechanism, our solution does not limit the flexibility and power of mobile agents' paradigm. The mobile agent does not need to interact with its home base during its whole journey. Besides, neither traces of intermediate execution states are required nor are more burdens put on agents' developer. In our solution, we have relaxed the need to provide the Third Trusted Party with an accurate proof of malicious acts. Capabilities of the Third Trust Party are enlarged. Consequently, an inexpensive protocol is derived to detect the suspicion character of hosts based on irregularities that could be observed on times of executions. These times are readjusted by the mediating activity (i.e., Third Trusted Party) to reflect hosts' referential power of processing and load. This will give better accuracy. Third Trusted Party elaborates more developed analysis. When more confidence is gained on the claim, the analysis will use a process of truth enforcement. Alternatively, when the claim is no more than allegations, the process will stop. Truth enforcement tries to flood the suspicious host with some forms of interrogating questions. However, the interrogation must not be **blind** and floods both the interrogation parties. The distinction is possible if the flooding task is suspicious' host pertinent information centric. Consequently, the Third Trusted Party can proceed normally as long as it can afford the management of **questions/responses**, and will switch to the average calculation otherwise.

4. A New Strategy Dealing with Malicious Hosts' Problem

We propose a new strategy dealing with the problem of malicious hosts. The defined strategy has two stages. During the strategy's first stage, we will develop an off-line property preserving protocol to detect the low suspicious character of any host. The protocol makes the processing power of a host proportional to its load. This goal is achieved by ignoring external data time acquisition (i.e., *black statements*). In its final step, the protocol will look for incoherencies that may occur between time fragments spent at each host with respect to the ones spent at first hop. Thereafter, these incoherencies are sent to the Third Trusted Party for depth evaluation. When better claim confidence is gained, the truth enforcement activity takes over. Its goal is to push the suspicious host to the limits of its full capacities. Crossing suspicious host scarce resources boarder of practical management will hopefully procreate a situation where the host is in full disarray. The host disarray, illustrated by incoherencies in its responses' log, is considered as the confirming proof of malicious acts. The next figure tries to depict the essence of agent's execution in a host, principal element for the comprehension of the strategy functionality:

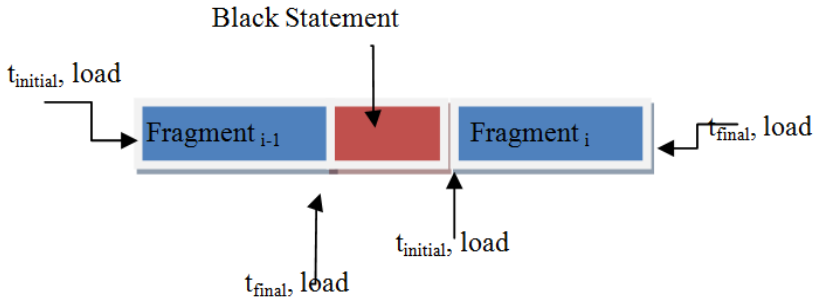

Figure 1: Agent's Code Fragmentation

4.1. Suspicious Character Detection Protocol

Our protocol requires only a simple model of mobile agent paradigm. Hence, the technology underlying the implementation of the mobile agent model provides only cryptographic mechanisms, which can protect the agent's confidentiality and integrity during transit. Besides, each net host must communicate to the Third Trust Party service (i.e., mediation), its referential power of processing and the corresponding load. The communication must take place before a host is made available.

We are considering a scenario in which the mobile agent performing a particular task is roving from host to host starting at a home base. Eventually, the agent will terminate and the results are delivered to its owner at the home base. Then, the agent owner at the home base checks for suspicious acts, and forwards them for deeper investigations to the mediating service. We will explain the protocol steps in every stage by describing tasks fulfilled when the home base or an executing host is involved. Protocol's steps are explained below:

1. The origin host ciphers the mobile agent with the public key of the first hop host and moves it forward.
2. When receiving the mobile agent, the host deciphers it and saves into its state the signed arrival time and current load before running it. When the execution reaches the first boundary of a black statement, the host again will save the signed current time and load to agent's state. The same task is performed upon black statement termination as well as just before the next jump is initiated. When a jump is encountered, the actual host ciphers the mobile agent with the next hop public key and sends it to its destination. When the mobile agent finishes, it will return to the home base.
3. The origin host deciphers the mobile agent and starts suspect acts verification. Just as explained before, the first hop execution time is taken to be as the referential model of time. Therefore, subsequent hops' fragments of time are readjusted to reflect the actual processing power of the referential host. The essence of the readjustment task is to bring a host's fragments of times to the corresponding segments' load of the referential host .i.e., the referential model of time. Recall that, the processing power of a host is made proportional to its load by ignoring black statements times. Hence, the readjustment factors are computed on the basis of relevant loads. Concretely, the new time value of a fragment i of a host is recalculated as in the following formula:

$$Readjusted\_Fragment_i\_time = \left\{ \frac{(referential\_host\_fragment_i\_average\_load)}{(host\_fragment_i\_average\_load)} \right\} \times host\_fragment_i\,(t_{initial} - t_{final})$$

An ordinary host (i.e., a one which is different from the referential host) is declared to be suspicious when at least one of its readjusted fragments of times is greater than its relative referential host's fragment of time. However, the referential host would be declared to be suspicious when each of its fragments of times is greater to any other host relative readjusted fragment of time. When at least one host is declared to be suspicious, the agent owner builds a self signed proof ground and sends it to the mediation entity. Proof ground is composed of relevant hosts' signed fragments of times, and corresponding loads. There are two distinct cases worth noticing here, depending upon the identity of the suspicious host. When the referential host (i.e., first hop host) is declared to be suspicious, the proof ground must include all hops relative information. Otherwise, only suspicious and referential hosts' relative information is required. The signed proof ground will be analyzed in the mediation activity by the Third Trusted Party.

4.2. Mediation Protocol

The mediation task is performed by Third Trusted Party. Its goal is to check whether or not a claim of an agent owner is just an allegation or the malicious character of a host is confirmed. The Third Trusted Party proceeds in a progressive manner; it starts by asserting the veracity of the referential host. Then, it verifies the soundness of the claim. When more confidence is gained, the Third trusted Party conducts the ultimate stage (i.e. Truth enforcement). Conducted steps are developed according to their natural sequence:

4.2.1. Asserting Referential Host Veracity

Upon receiving a signed complaining message, i.e., a proof ground from an agent owner, the Third Trusted Party starts by asserting the veracity of the referential host. We must notice that in the referential host, an agent execution is supposed to be safe since it excludes surreptitious modifications. Therefore, its duration is expected to be minimal. This represents a crucial key for referential host veracity checking. Based on this assumption, the process of veracity checking starts by readjusting other hosts' fragments of times, using the same formula as in the former paragraph. Then, it brings each host's fragment of time to real time by using the host's referential load and referential power of processing. After that, it selects the resulting minimal duration real time of all hosts. The referential host's status is declared to be the one of referential model of execution (i.e.; trusted) in case where the sum of minimal durations of real times is equal to the referential host's total duration of real times. More accurate mathematical expressions are given below where order steps should be preserved:

1. For each host different of the referential host do

   For each $fragment_i$ do

$$Readjusted\_Fragment_i\_time \leftarrow \left\{ \frac{referential\_host\_fragment_i\_average\,load}{host\_fragment_i\_average\_load} \right\} \times host\_fragment_i\,(t_{initial} - t_{final})$$

2. For first hop Host Only

For each $Fragment_i\_time$ do

$$Readjusted\_referential\_host\_fragment_i\_time \leftarrow referential\_host\_fragment_i\_time$$

For each host including the referential host do

For each $Readjusted\_fragment_i\_time$ do

$$Real\_fragment_i\_time \leftarrow \left\{ \frac{host\_referential\_load}{host\_fragment_i\_average\_load} \right\} \times host\_referential\_Power\_of\_Processing \times Readjusted\_Fragment_i\_time$$

3. For each $fragment_i\_time$ of all hosts

Select the minimal $Real\_fragment_i$ time

If $\left(\sum minimal\ Real\_fragment_i\_time = \sum referential\_host\_Real\_fragment_i\_time \right.$

Then the referential host is truly the model of execution

Figure 2: Referential Host Veracity Checking Process

4.2.1. Soundness of the Claim

Here, we verify the soundness of the claim as previously done in the suspicious character detection protocol formula. More confidence is gained on the claim when at least one host's real fragment of time is greater than the real time of the relative referential host's fragment, or all referential host real fragments of times are superior to all others hosts' real fragments of times. Then, the Third Trusted Party conducts the ultimate stage (i.e. Truth Enforcement). Otherwise, the claim is found to be just an allegation, and the Third Trusted Party may increase the severity log relative to the agent owner, send him a warning or decide to punish him if a threshold is reached.

4.2.3. Truth Enforcement

During this stage, the process of truth enforcement is activated. It starts by requesting the suspicious host pertinent information (e.g., its price). Thereafter, it will manage the questions/responses dialogue formed from first order logic propositions. We must notice that the Third Trusted Party knows in advance the true outcomes of its propositions. This dialog will go through sessions attached to two different operating modes. In the first operating mode, session's management purposes are:

1. Generating suspicious host price centric questions (i.e., first order logic propositions) at random where any former questions are reinserted.
2. Detecting any suspicious host attempt to remember former questions.
3. Detecting incoherencies among sessions' responses.

The first operating mode will last as long as the Third Trusted Party can manage required resources efficiently and responses are coherent (when responses are incoherent the suspicious host is declared to be malicious and the mediation protocol stops). However, upon this frontier, the Third Trusted Party moves onto the second

operating mode. Then, the Third Trusted Party would favor truth enforcement initiation whenever it had noticed the suspicious host's attempt to remember any previously asked question at any moment of the dialog. Otherwise, the host is declared to be honest. Truth enforcement process tries to push rapidly the suspicious host to the limit of clogged state. Hopefully, before becoming faulty, the suspicious host will not be able to give coherent answers. During this stage of the second operating mode, the Third Trusted Party sessions performs the following actions to enforce truth:

1. Generating suspicious host price centric questions (i.e., first order logic propositions) at random where the first mode of operating questions are reinserted.
2. Supervising responses will focus only on the average of their results. We should notice now that the Third Trusted Party cannot afford individual response supervision due to the lack of resources.
3. Terminating this second operating mode when the average of the responses results is not as expected. Thereafter, the host is declared to be malicious.

Previously, the former section has sketched the essence of our mediating protocol performed by the Third Trusted Party. As the protocol's finest part aims at pushing the suspicious host into full disarray progressively, we will discuss practical questions related to this particular aspect. Among questions of great importance are propositions generation, sessions' sizes, randomness of session's propositions ordering, and how to express previous propositions.

Proposed answers would clarify why the suspicious host could be flooded whereas the Third Trusted Party won't. We will start by considering sessions' sizes. In its initial state, the protocol could use a hyper geometric function to set randomly the first session's size. Subsequent sessions' sizes must increase to manage the set of previous propositions as well as the relevant new ones. Obviously, the propositions generation mechanism is of prime importance. A systematic generation requires propositions which are suspicious host price centric. In practice, a centered price interval which includes implicitly a range of values set equals to the session's size is enough. Hence, expressing a proposition could be for instance as simple as: "is (price – a lower interval constant value) equal to the real value of the difference". Propositions' generation process must enclose a germ for detecting hosts attempts to remember previously asked questions. Consequently, previous propositions must be expressed differently, in a way which can make remembering attempts Third Trust Party perceivable. For this purpose, we will use a reserved key word denoting the right hand term of a proposition. Then, we can incorporate previously asked propositions as simply as using a key notation referring to the second member of a proposition as in the following example: "is (price – Former_Session[i].Right_Hand_term) {=, >, <} to the real value of the difference". Clearly, to randomly rearrange propositions' ordering, we can use any standard method that could be found in the computer science literature.

5. Robustness Analysis of Suspicious Character Detection Protocol

In essence, our approach [12] makes the processing of each host proportional to its load. The technique consists of ignoring catalog querying time. The suspicious character detection protocol considers the first host as the referential model of execution. Therefore, we bring other computing hosts times to its corresponding

computing speed. Not only, has this choice discarded the need to use managing components for both the general time reference and the time estimation, but it reflects a basic reality. Reality speaks for ‘’first host has no fugitive action other than moving the roving agent to a different next place’’. First host’s safe execution excludes surreptitious modifications, and thus, it is expected to be minimal. Essentially, a fugitive action must leave the state of the agent coherent.

5.1. Working Example

We devote a special care for validating our approach. We have selected an example providing an opportunity to exploit the power of the proposed solution. Our approach had cut off the requirements complexity of Vigna’s and Esparsa’s works. Therefore, Vigna’s example is extended to encompass our approach particularities. For instance, instead of two shops, the number is enlarged to be three. This larger number of shops enables the particular scenario where first hop host could modify mobile agent itinerary. In the simple chosen mobile agent application, we consider a user at site home.sweet-home.com who wants to buy a home video of Tarantino’s Pulp Fiction movie. Therefore, he dispatches his agent to a site called agents.virtualmall.com dedicated to maintain a directory of electronic shops. Once there, the agent performs directory query for sites offering home video. Then, the mobile agent visits the provided sites. At each site the agents contacts the local catalog service to determine the current price of the Pulp Fiction home video. When all prices have been collected, the agent identifies at home the best offer and, if the best price is less than a specified amount —say, twenty dollars—the mobile agent goes to the selected site and buys the home video.

5.2. Technical Considerations

Our roving agent carries with it visited host’s signed computing times and loads. Catalog querying frontiers represents borders between fragments of times. The test bed java program had been carried on **Lunix** Pentium III 866 MHZ machine. The common question people ask is “How fast does a Program ‘’ run on a Machine’’? In our earlier discussion, we assumed this question could be answered with perfect accuracy. It turns out that this problem is surprisingly complex. There are many factors that can vary from one execution of a program to another. Computers do not simply execute one program at a time; they continually switch from one process to another, executing some code on behalf of one process before moving on to the next. The exact scheduling of processor resources for one program depends on such factors as the number of users sharing the system, the network traffic, and the timing of disk operations. The access patterns to the cache depend not just on the references made by the program we are trying to measure, but on those of other processes executing concurrently. If our proposed protocol copes easily with the number of the programs currently in use, it ignores disk operations, and considers the cache silent which is in fact, the real case in our context. Unfortunately, we were unable to get values of the computer load during the test. System’s Load is a necessary element for the good work of our protocol. To overcome our programming shortcoming, we will use a statistical approach. Specifically, hypothesis Test for the Difference between Two Means called **two sample t_test**.

For the purpose of validation, different scripts of java programs had been developed. Mainly, scripts iterate on mobile agent execution one hundred of times. The different

images of our mobile agent had encompassed both safe and unsafe (effectively malicious) hosts. Thereafter, we had analyzed and plotted simulation results as histograms for better readability. As iterations range is not important, execution times fragments have been sorted. Next are times fragments graphs for the different images of the mobile agent.

5.2. Simulation Results and Statistical Interpretation

The results as graphs show simulation outputs in different situations. These results are obtained from the mobile agent different images executions. Images stand for referential model host, safe and malicious hosts. Next are times fragments graphs of the different images executions of the mobile agent:

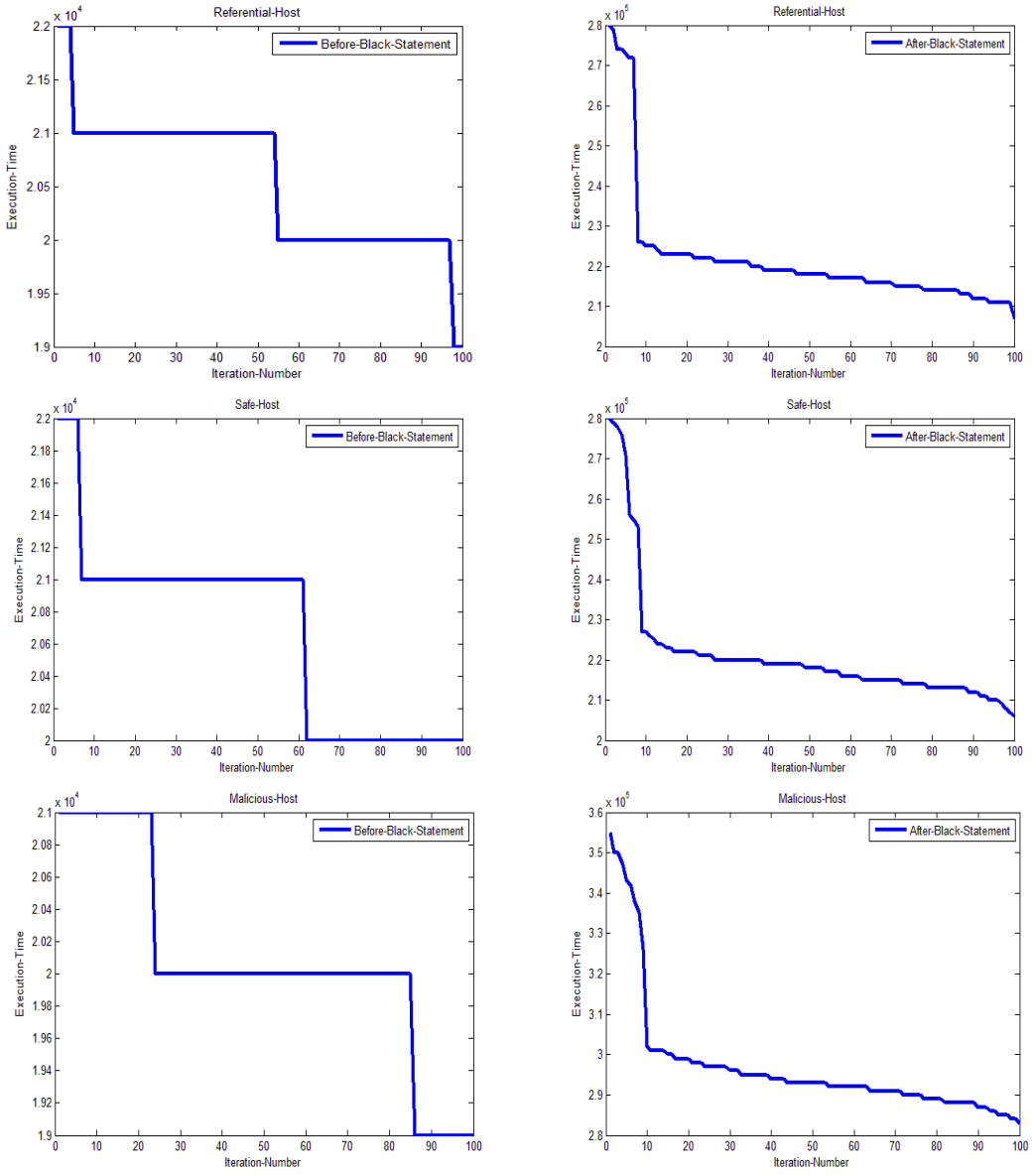

Figure 3: Hosts' Fragments of Execution Times Graphs

Numerically speaking, the crucial graphs' information is summarized by the following table, pointing out means **(μ)** and Standard Deviations **(SD)** with respect to each host of the example:

Table 1: Agent Images' Execution Times Means and Standard Deviations

Time In Nano Seconds	Referential_Host	Safe_Host	Malicious_Host
BeforeBlack Statement	μ =20500 SD=508	μ =21000 SD=508	μ =20000 SD=508
AfterBlack Statement	μ =243500 SD=508	μ =243000 SD=508	μ =319000 SD=508

Comparing graphs, especially the elements of the table above shows a case where the third host had acted maliciously. In fact, its second mean fragment time of execution is much bigger than the mean time of the referential host second fragment of time. This is a case where the protocol detects the weak suspicious character of the third host. The image of the third host reflects a situation of abnormal winning because the state of the mobile agent had been modified (requiring more time) to make third host price best offer.

Let's now explain and perform the statistical process of validation. Our test involves formulating and testing **hypotheses**, assertions that are falsifiable using a test of observed data. The **null hypothesis** typically proposes a general or default position, such as that there is no relationship between two measured phenomena. It is typically paired with a second hypothesis, the **alternative hypothesis** which asserts a particular relationship between the phenomena. Recall that our suspicious character protocol starts by asserting the veracity of **the referential host** .i.e., model of execution. Thereafter, it will analyze its difference with respect to other journey hosts.

Data collected during the experiment standing for a series with the same size of samples show an equal standard deviation of the measured times of execution. Manifestly, we suggest that two relevant measured times of executions would tie whenever their corresponding distance .i.e., absolute difference is less than the unique standard deviation. Hence, the appropriate statistical hypothesis test is the **Hypothesis Test for the Difference between Two Means** called **two-sample t-test**. This approach consists of four steps: (1) state the hypotheses, (2) formulate an analysis plan, (3) analyze sample data, and (4) interpret results.

Every hypothesis test requires the analyst to state a null hypothesis and an alternative hypothesis. The hypotheses are stated in such a way that they are mutually exclusive. That is, if one is true, the other must be false; and vice versa. The table below shows three sets of null and alternative hypotheses. Each set makes a statement about the distance *d* between the mean of one population μ_1 and the mean of another population μ_2.

Table 2: Set of Null and Alternative Hypotheses

Set	Null hypothesis	Alternative hypothesis	Number of tails
1	$\| \mu_1 - \mu_2 \| = d$	$\mu_1 - \mu_2 \neq d$	2
2	$\mu_1 - \mu_2 \geq d$	$\mu_1 - \mu_2 < d$	1
3	$\mu_1 - \mu_2 \leq d$	$\mu_1 - \mu_2 > d$	1

The first set of hypotheses **(Set 1)** is an example of a two-tailed test, since an extreme value on either side of the sampling distribution would cause a researcher to reject the null hypothesis. The other two sets of hypotheses **(Sets 2 and 3)** are one-tailed tests, since an extreme value on only one side of the sampling distribution would cause a researcher to reject the null hypothesis.

The analysis plan describes how to use sample data to accept or reject the null hypothesis. It should specify the following elements:

1. Choosing the significance level.
2. Using the two-sample t-test to determine whether the difference between means found in the sample is significantly different from the hypothesized difference between means.

The analyzing of sample data finds the standard error, degrees of freedom, test statistic, and the P-value associated with the test statistic.

1. Computing the standard error **(SE)** of the sampling distribution by: SE = **sqrt**$[(s_1^2/n_1) + (s_2^2/n_2)]$ where s_1 is the standard deviation of sample1, s_2 is the standard deviation of sample 2, n_1 is the size of sample 1, and n_2 is the size of sample 2.
2. Calculating the degrees of freedom (DF) is:

   $DF = (s_1^2/n_1 + s_2^2/n_2)^2 / \{[(s_1^2 / n_1)^2 / (n_1 - 1)] + [(s_2^2 / n_2)^2 / (n_2 - 1)]\}$
3. Defining the test statistic as t-score (t) defined by the following equation: $t = [| (x_1 - x_2) | - d] / SE$ where x_1 is the mean of sample 1, x_2 is the mean of sample 2, d is the hypothesized difference between population means, and SE is the standard error.
4. Establishing by calculator the P-value which is the probability of observing a sample statistic as extreme as the test statistic. Since the test statistic is a t-score, use the t Distribution Calculator to assess the probability associated with the t-score, having the degrees of freedom computed above.

The results interpreting would see if the sample findings are unlikely, given the null hypothesis, we reject the null hypothesis. Typically, this involves comparing the **P-value** to the significance level, and rejecting the null hypothesis when the **P-value** is less than the significance level.

Back to our real test, we remark that **SE** and **DF** are the same since the standard deviation of our samples is the same i.e. 508. They have been calculated just once, and their values are respectively 71.842049 and 198. The following table states the hypothesis for referential host veracity checking:

Table 3: Working Example Null Hypothesis

HostToHost	Before_Black_Statement	After_Black_Statement
Referential_Host To Safe_Host	$\mid \mu_{\text{Referential}} - \mu_{\text{Safe}} \mid = ?500 \leq 508$ meaning that they are tie	$\mid \mu_{\text{Referential}} - \mu_{\text{Safe}} \mid = ?500 \leq 508$ meaning that they are tie
Referential_Host To Malicious_Host	$\mid \mu_{\text{Referential}} - \mu_{\textit{Malicious}} \mid = ?500 \leq 508$ meaning that they are tie	$\mid \mu_{\text{Referential}} - \mu_{\textit{Malicious}} \mid = ?500 \leq 508$ meaning that they are tie

When applying values, the corresponding resulting **t-scores** and **P-values** are summarized below:

Table 4: Null Hypothesis t-scores and P-Value

ReferentialHost To aHost	Before_Black_Staement		After_Black_Statement	
	t-score	P-Value	t-score	P-Value
Referential_Host To Safe_Host	0	1	0	1
Referential_Host To Malicious_Host	0	1	1043.9569	0

Except the null hypothesis of the "**After_Black_Statement**'' where the **P-value** is equal to zero, the referential host can be assured certainly of a partial minimal execution time (probability equal to one speaks for certainty). If we can prove that $\mu_{\text{Malicious}} - \mu_{\text{Referential}} > 508$ let say 509, we can state formally that the referential host is truly a referential model of execution and that malicious host has acted truly maliciously. It is a one tailed test where t-score is ((75500 - 509) /71.842049) = 1043.8316. The calculator tells us that: $P(t < 1043.8316)=0.0$. Thus, the P-value is: $P(t > 1043.8316)= 1 - P(t < 1043.8316) = 1$.

The fact that simulation has been carried on the same computer de facto overrides the need to carry out the required fragments of real times calculation. Hence, our former conclusion .i.e., third host has acted maliciously is still valid. Finally, we should notice that the presented example reflecting malicious acting is not specific, Rather, it exemplifies any situation were malicious acts are performed. Therefore, we could conclude that simulation has comforted the protocol on its ongoing goal, and the protocol robustness has effectively been proved.

6. Conclusion

This paper has designed a protocol which proceeds incrementally to determine whether or not a host is malicious. The paper has motivated, explored, and expressed solutions based on a new research direction with respect to the malicious hosts' problem. To our protocol's

proposal, we have joined a section to answer related practical questions. The experiment made has proved the effectiveness of the decisions we have made. The valuable weaker hints, which are easily collected, have made the suspicious character of any host verification rather an easy task. Very soon, we will experiment the truth enforcement process, and we hope that it will determine unambiguously the malicious character of a host as we are expecting.

References

[1] S. Dobrev and Al. ''Mobile Search for a Black Hole in an Anonymous Ring''. Proceeding of the 15th International Conference on Distributed Computing **(2001)** London, UK.

[2] Robert S. Gray, David Kotz, and Ronald A. Peterson, jr. ''Mobile-Agents Versus Client/Server Performance : Scalability in Information-Retrieval Task. Technical Report TR 2001-386 **(2001)** January 30, Darmouth College Computer Science.

[3] A. Barak and Y. Kornatzky. ''Design Principles of Operating Systems for large Scale Multicomputers''. IBM Research Division, T.J. Watson Research Center. New York. RC13220(#59114) **(1987)**

[4] T. Sander and Christian F.T Tshudan. ''On Software Protection via Function Hiding'', submitted to the 2 International Workshop on Information Hiding.

[5] T. Sander and Christian F.T Tshudan. ''Towards Mobile Cryptography''. IEEE Symposium on Security and Privacy **(1988)** May.

[6] Fritz. Hohl. ''Time Limited Black Box Security: Protecting Mobile Agents from Malicious Hosts''. Springer Verlag **(1998)** Citseer.IST. Scientific Literature Digital Library.

[7] Lora L. Kassab and Jeffry Voas. ''Agents Trustworthiness''. Ecoop Workshop, 1998:300 **(1998)**

[8] Giovanni Vigna. ''Protecting Mobile Agents Through Tracing''. In Proceeding of the 3rd Ecoop Workshop on Mobile Objects Systems, Jyvalskyla, Finland **(1997)**

[9] Uwe G. Wilhem, Sebastian M. Staamann, and Levente Buttyan. '' A Pessimistic Approach to Trust in Mobile Agent Platforms''. IEEE Internet Computing September-October 2000. http://computer.org/internet/ **(2000)**

[10] O. Esparza, M. Soriano, J.L. Munoz, and J. Forné. ''A Protocol for Detecting Malicious Hosts Based on Limiting the Execution Time of Mobile Agents''. In IEEE Symposuim on Computers and Communications-ISCC'2003 **(2003)**

[11] Carles Garrigues Olivella. P.H.D. Thesis, ''Contribution To Mobile Agent Protection from Malicious Hosts'', Universitate Autonoma de Barcelona **(2008)** June.

[12] M. Chihoub ''Malicious Hosts Detection Through Truth Powered Approach'', In Proceeding of Third International annual Workshop on Digital Forensics and Incident Analysis, IEEE Computer Society, Malaga, Spain **(2008)** October 9.